An unsupervised algorithm for learning Lie group transformations

Jascha Sohl-Dickstein[1, 5, 6]
Ching Ming Wang[2, 4, 5, 6]
Bruno Olshausen[2, 3, 5, 6]
[1]Biophysics Graduate Group.
[2]School of Optometry
[3]Helen Wills Neuroscience Institute
[4]earthmine inc.
[5]Redwood Center for Theoretical Neuroscience
[6]University of California at Berkeley, Berkeley, CA

Keywords: Lie group, natural video, transformations, scene statistics, dynamical systems

Abstract

We present several theoretical contributions which allow Lie group, or continuous transformation, models to be fit to large high dimensional datasets. We then demonstrate training of Lie group models on natural video. Transformation operators are represented in their eigen-basis, reducing the computational complexity of parameter estimation to that of training a linear transformation model. A transformation specific "blurring" operator is introduced that allows inference to escape local minima via a smoothing of the transformation space. A penalty on traversed manifold distance is added which encourages the discovery of sparse, minimal distance, transformations between states.

Both learning and inference are demonstrated using these methods for the full set of affine transformations on natural image patches. Transformation operators are then trained on natural video. It is shown that the learned video transformations provide a better description of inter-frame differences than the standard motion model, rigid translation.

1 Introduction

A fundamental problem in vision is to find compact descriptions for how images change over time. Such descriptions may provide clues to the representation used in the brain [Einhauser et al., 2002, Olshausen et al., 2007, Cadieu and Olshausen, 2008], and they could lead to more efficient video compression [Wiegand et al., 2003] and motion estimation algorithms. Better characterization of the statistics of natural video would also allow for the generation of more natural, controlled stimuli for use in psychophysical and neurophysiological experiments [Victor et al., 2006]. Finally an understanding of dynamics could lead to better methods for extracting visual invariants - both *form invariants*, where an object retains its form under changes in position, lighting or occlusion [LeCun et al., 2004, Wallis and Rolls, 1997, Serre et al., 2007], and *transformation invariants* [Cadieu and Olshausen, 2008], where the same transformation is applied to an object independent of its form.

Motivated by the problem of recognizing form invariants, [Rao and Ruderman, 1999] introduced the idea of learning a Lie, or continuous transformation, group representation of the dynamics which occur in the visual world. The Lie group is built by first describing all infinitesimal transformations which an image may undergo. The full group is then generated from all possible compositions of those infinitesimal transformations, which allows for transformations to be applied smoothly and continuously. A large class of visual transformations, including all the affine transformations, intensity changes due to changes in lighting, contrast changes and spatially localized versions of all the preceding, can be described simply using Lie group operators (although other transformations – for instance moving occlusion boundaries – cannot be easily described). In [Miao and Rao, 2007, Rao and Ruderman, 1999], the Lie group operators were trained on image sequences containing a subset of affine transformations. [Memi-

sevic and Hinton, 2010] trains a second order restricted Boltzmann machine on pairs of frames, an alternative technique which also shows promise for capturing temporal structure in video.

Unfortunately, despite the simplicity and power of the Lie group representation, training such a model is difficult, partly due to the high computational cost of evaluating and propagating learning gradients through matrix exponentials. Previous work [Olshausen et al., 2007, Rao and Ruderman, 1999, Miao and Rao, 2007] has approximated the full model using a first order Taylor expansion, reducing the exponential model to a linear one. While computationally efficient, a linear model approximates the full exponential model only for a small range of transformations. This can be a hinderance in dealing with real world data, which oftentimes contain a large range of changes between pairs of video frames. Note that in [Miao and Rao, 2007], while the full Lie group model is used in inferring transformation parameters, only its linear approximation is used during learning. [Culpepper and Olshausen, 2010] work with a full exponential model, but their technique requires performing a costly eigendecomposition of the effective transformation operator for each sample and at every learning or inference step.

Another hurdle one encounters is that the inference process, which computes transformation coefficients given a pair of images, is highly non-convex with many local minima. This problem has been extensively studied in image registration, stereo matching and the computation of optic flows. For a certain set of transformations (translation, rotation and isotropic scaling), [Kokiopoulou and Frossard, 2009] showed that one could find the global minima by formulating the problem using an overcomplete image representation. For arbitrary transformations, one solution is to initialize inference with many different coefficient values [Miao and Rao, 2007]; but the drawback is that the number of initial guesses needed grows exponentially with the number of transformations. Alternatively, [Lucas and Kanade, 1981, Arathorn, 2002, Vasconcelos and Lippman, 1997, Black and Jepson, 1996] performs matching with an image pyramid, using solutions from a lower resolution level to seed the search algorithm at a higher resolution level. [Culpepper and Olshausen, 2010] used the same technique to perform learning with Lie Group operators on natural movies. Such piecewise coarse-to-fine schemes avoid local minima by searching in the smooth parts of the transformation

space before proceeding to less smooth parts of the space. However, they require that the coarsening the transformation corresponds to spatial blurring. As we show here, it is also possible to smooth the transformation space directly, resulting in a robust method for estimating transformation coefficients for arbitrary transformations.

In this work we propose a method for directly learning the Lie group operators that mediate continuous transformations, and we demonstrate the ability to robustly infer transformations between frames of video using the learned operators. The computational complexity of learning the operators is reduced by re-parametrizing them in terms of their eigenvectors and eigenvalues, resulting in a complexity equivalent to that of the linear approximation. Inference is made robust and tractable by smoothing the transformation space directly, which allows for a continuous coarse-to-fine search for the transformation parameters. Both learning and inference are demonstrated on the full set of affine transformations using a Lie group framework. The same technique is then used to learn a set of canonical transformations describing changes between frames in natural movies. Unlike previous Lie group implementations [Rao and Ruderman, 1999, Miao and Rao, 2007, Olshausen et al., 2007, Culpepper and Olshausen, 2010], we demonstrate an ability to work simultaneously with multiple transformations and large inter-frame differences during both inference and learning.

2 Continuous Transformation Model

As in [Rao and Ruderman, 1999], we consider the class of continuous transformations described by the first order differential equation

$$\frac{\partial y\left(s\right)}{\partial s} \;=\; A\, y\left(s\right), \tag{1}$$

with solution

$$y\left(s\right) = e^{As} y\left(0\right) = T\left(s\right) y\left(0\right). \tag{2}$$

Here $A \in \mathcal{R}^{N\times N}$ is an infinitesimal transformation operator and the generator of the Lie group; $s \in \mathcal{R}$ is a coefficient which controls its application; $T\left(s\right) = e^{As}$ is a matrix exponential defined by its Taylor expansion and $y\left(s\right) \in \mathcal{R}^{N\times 1}$ is the signal $y\left(0\right)$ transformed by A to a degree controlled by s.

The goal of this paper is to use transformations of this form to model the changes between adjacent frames $x^{(t)}, x^{(t+1)} \in \mathcal{R}^{N\times 1}$ in video. That is, we seek to find the model parameters A (adapted over an ensemble of video image sequences) and coefficients $s^{(t)}$ (inferred for each pair of frames) that minimize the reconstruction error

$$E = \sum_t \left|\left|x^{(t+1)} - T\left(s^{(t)}\right) x^{(t)}\right|\right|_2^2. \tag{3}$$

We will later extend this to multiple transformations.

2.1 Eigen-decomposition

To derive a learning rule for A, it is necessary to compute the gradient $\frac{\partial e^{As}}{\partial A}$. Naively this costs $O\left(N^6\right)$ time [Ortiz et al., 2001] ($O\left(N^2\right)$ operations per element, and N^4 elements), making it computationally intractable for many problems of interest. However, this computation reduces to $O\left(N^2\right)$ (the same complexity as its linear approximation) if A is rewritten in terms of its eigen-decomposition

$$A = U\Lambda U^{-1} \tag{4}$$

and learning is instead performed directly in terms of U and Λ.[1] $U \in \mathcal{C}^{N\times N}$ is a complex matrix consisting of the eigenvectors of A, $\Lambda \in \mathcal{C}^{N\times N}$ is a complex diagonal matrix holding the eigenvalues of A, and U^{-1} is the inverse of U. The matrices must be complex in order to facilitate periodic transformations, such as rotation. However, note that U need not be orthonormal. The benefit of this representation is that

$$e^{U\Lambda U^{-1}s} = I + U\Lambda U^{-1}s + \frac{1}{2}U\Lambda U^{-1}U\Lambda U^{-1}s^2 + \ldots = Ue^{s\Lambda}U^{-1} \tag{5}$$

where the matrix exponential of a diagonal matrix is simply the element-wise exponential of its diagonal entries. This representation therefore replaces the full matrix exponential by two matrix multiplications and an element-wise exponential.

2.2 Adaptive Smoothing

Unfortunately, in general the reconstruction error described by Equation 3 is highly non-convex in s and contains many local minima. To illustrate, the red solid line in Figure

[1] This change of form enforces the restriction that A be diagonalizable.

1 plots the reconstruction error for a white-noise image patch shifted by three pixels to the right as a function of transformation coefficient s for a generator A corresponding to left-right translation. It is clear that performing gradient based inference on s for this error function would be problematic.

To overcome this problem, we propose an alternative transformation, motivated by modern image matching algorithms [Arathorn, 2002, Vasconcelos and Lippman, 1997, Black and Jepson, 1996, Lucas and Kanade, 1981], that adaptively smooths the error function in terms of the transformation coefficient. This is achieved by averaging over a range of transformations using a Gaussian distribution for the coefficient values

$$\begin{aligned} T(\mu,\sigma) &= \int_{-\infty}^{\infty} T(s)\,\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{||s-\mu||^2}{2\sigma^2}} ds \\ &= U e^{\mu\Lambda} e^{\frac{1}{2}\Lambda^2\sigma^2} U^{-1} \end{aligned}$$

and replacing $T(s)$ with $T(\mu,\sigma)$ in Equation 3, which is then minimized with respect to both μ and σ.[2]

Instead of the single best s that minimizes E, inference using $T(\mu,\sigma)$ finds a Gaussian distribution over s, effectively blurring the signal along the transformation direction given by $A = U\Lambda U^{-1}$. In the case of translation, for instance, this averaging over a range of transformations blurs the image along the direction of translation. The higher the value of σ, the larger the blur. Under simultaneous inference in μ and σ, images are matched first at a coarse scale, and the match refines as the blurring of the image decreases.

To illustrate the way in which the proposed transformation leads to better inference, the dotted lines in Figure 1 shows the reconstruciton error as a function of μ with different values of σ. Note that, by allowing σ to vary, steepest descent paths open out of the local minima, detouring through coarser scales.

[2]This can alternatively be seen as introducing an additional transformation operator, this one a smoothing operator

$$A_{smooth} = \frac{1}{2} U\Lambda^2 U^{-1}, \tag{6}$$

with coefficient $s_{smooth} = \sigma^2$. A_{smooth} smooths along the transformation direction given by $A = U\Lambda U^{-1}$.

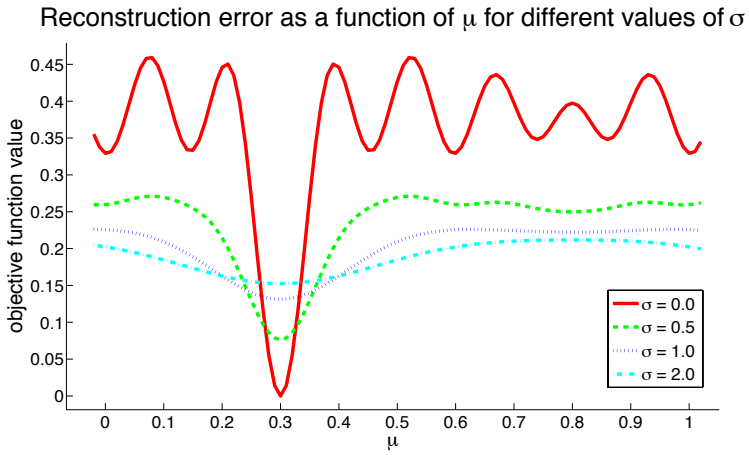

Figure 1: Reconstruction error (Equation 3) as a function of μ for different values of σ. In this case the target pattern $x^{(t+1)}$ has been translated in one dimension relative to an initial white noise pattern $x^{(t)}$, and the operator A is the one-dimensional translation operator. Local minima in terms of μ can be escaped by increasing σ.

3 Multiple Transformations

A single transformation is inadequate to describe most changes observed in the visual world. The model presented above can be extended to multiple transformations by concatenating transformations in the following way:

$$T_{\text{multi}}(\mu, \sigma) = T_1(\mu_1, \sigma_1) T_2(\mu_2, \sigma_2) \ldots = \prod_k T_k(\mu_k, \sigma_k) \tag{7}$$

$$T_k(\mu_k, \sigma_k) = U_k e^{\mu_k \Lambda_k} e^{\frac{1}{2}\Lambda_k^2 \sigma_k^2} U_k^{-1} \tag{8}$$

where k indexes the transformation. Note that the transformations $T_k(\mu_k, \sigma_k)$ do not in general commute, and thus the ordering of the terms in the product must be maintained.

Because of the fixed ordering of transformations and due to the lack of commutativity, the multiple transformation case no longer constitutes a Lie group for most choices of transformation generators A. Describing the group structure of this new model is a goal of future work. For present purposes we note that for several obvious video transformations - affine transformations, brightness scaling, and contrast scaling - the accessible transformations are not restricted by the model form in Equation 7, though the choice of coefficient values μ_k for a transformation may depend heavily on the order of terms in the product.

4 Regularization via Manifold Distance

In order to encourage the learned operators to act independently of each other, and to learn to transform between patches in the most direct way possible, we penalize the distance through which the transformations move the image patch. If the initial image patch is $y_1(0)$, this distance can be expressed as

$$d\left(T_{\text{multi}}\left(\mu,\sigma\right)y_1(0)\right) = \sum_k d\left(T_k\left(\mu_k,\sigma_k\right)y_k(0)\right) \tag{9}$$

where $y_k(0) = \prod_{m<k} T_m\left(\mu_m,\sigma_m\right)y_1(0)$ is the image patch before application of transformation k. Assuming a Euclidean metric, the distance $d\left(T_k\left(\mu_k,\sigma_k\right)y_k(0)\right)$ traversed by each single transformation in the chain is

$$\begin{aligned} d\left(T_k\left(\mu_k,\sigma_k\right)y_k(0)\right) &= \int_{\tau=0}^{s_k} ||\dot{y}_k(\tau)||_2 d\tau & (10)\\ &= \int_{\tau=0}^{s_k} ||A_k y_k(\tau)||_2 d\tau & (11)\\ &= \int_{\tau=0}^{s_k} ||A_k e^{A_k\tau} y_k(0)||_2 d\tau & (12) \end{aligned}$$

Finding a closed form solution for the above integral is difficult, but it can be approximated using a linearization around $\tau = \frac{s}{2}$,

$$d\left(T_k\left(\mu_k,\sigma_k\right)y_k(0)\right) \approx \mu_k ||A_k e^{A_k\frac{\mu_k}{2}} y_k(0)||_2. \tag{13}$$

Since this penalty is applied individually to each operator, it also acts similarly to an $L1$ penalty on the path length of the transformations. This penalty will encourage travel between 2 points to occur via a path described by a single transformation, rather than by a longer path described by multiple transformations.

5 The Complete Model

The full model's objective function is

$$\begin{aligned} E\left(\mu,\sigma,U,\Lambda,x\right) = \quad & \eta_n \sum_t \left|\left|x^{(t+1)} - T_{multi}\left(\mu^{(t)},\sigma^{(t)}\right)x^{(t)}\right|\right|_2^2 \\ & + \eta_d \sum_t \sum_k \mu_k^{(t)} ||A_k e^{\frac{\mu_k^{(t)}}{2}A_k} x_k^{(t)}||_2 \\ & + \eta_\sigma \sum_t \sum_k (\sigma_k^{(t)})^2 \end{aligned} \tag{14}$$

(where a small L2 regularization term on $\sigma_k^{(t)}$ was found to speed convergence during learning). We used $\eta_n = 1$, $\eta_d = 0.005$, and $\eta_\sigma = 0.01$. Derivatives of the energy function are provided in the appendix. This model can be recast into a probabilistic framework in a straightforward fashion by setting

$$p(x, \mu, \sigma | U, \Lambda) \propto \exp\left(-E\left(\mu, \sigma, U, \Lambda, x\right)\right). \tag{15}$$

6 Inference and Learning

To find **U** and **Λ**, we employ a variational Expectation-Maximization type of optimization strategy, which iterates between the following two steps:

1. Load a fresh set of video patches x, then find optimal estimates for the latent variables $\hat{\mu}$ and $\hat{\sigma}$ while holding the estimates of the model parameters $\hat{U}$ and $\hat{\Lambda}$ fixed,

$$\hat{\mu}, \hat{\sigma} = \arg\min_{\mu,\sigma} E\left(\mu, \sigma, \hat{U}, \hat{\Lambda}, x\right). \tag{16}$$

2. Optimize the estimated model parameters $\hat{U}$ and $\hat{\Lambda}$ while holding the latent variables fixed,

$$\hat{U}, \hat{\Lambda} = \arg\min_{U,\Lambda} E\left(\hat{\mu}, \hat{\sigma}, U, \Lambda, x\right) \tag{17}$$

All optimization was performed using the L-BFGS implementation in Mark Schmidt's minFunc [Schmidt, 2009]. A similar optimization scheme has been used in [Lewicki and Olshausen, 1999].

There is a degeneracy in U_k, in that the columns (corresponding to the eigenvectors of A_k) can be rescaled arbitrarily, and A_k will remain unchanged as the inverse scaling will occur in the rows of U_k^{-1}. If not dealt with, U_k and/or U_k^{-1} will random walk into an ill conditioned scaling over many learning steps. As described in detail in the appendix, this effect is compensated for by rescaling the columns of U_k such that they have identical power to the corresponding rows in U_k^{-1}.

7 Experimental Results

7.1 Inference with Affine Transforms

To verify the correctness of the proposed inference algorithm, a set of known transformations were applied to natural image patches, and the transformation coefficients were inferred using a set of hand-designed operators. For this purpose a pool of 1000 11×11 natural image patches were cropped from a set of short BBC video clips. Each image patch was transformed by the full set of affine transformations simultaneously with the transformation coefficients drawn uniformly from the ranges listed below. [3]

Transformation Type	Range
horizontal translation	$\pm$ 5 pixels
vertical translation	$\pm$ 5 pixels
rotation	$\pm$ 180 degrees
horizontal scaling	$\pm$ 50 %
vertical scaling	$\pm$ 50 %
horizontal skew	$\pm$ 50 %

The proposed inference algorithm (Equation 16) is used to recover the transformation parameters. Figure 2 shows the fraction of the recovered coefficients which differed by less than 1% from the true coefficients. The distribution of the PSNR of the reconstruction is also shown. The inference algorithm recovers the parameters with a high degree of accuracy. The PSNR in the reconstructed images patches was also higher than 25dB for 85% of the transformed image patches. In addition, we find that adaptive blurring significantly improved inference, as evident in Figure 2a.

7.2 Learning Affine Transformations

To demonstrate the ability to learn transformations, we trained the algorithm on image sequences transformed by a single affine transformation operator (translation, rotation, scaling or skew). The training data we used were single image patches from the same BBC clips as in Section 7.1, transformed by an open source Matlab package [Shen, 2008] with the same transformation range used in Section 7.1.

[3] Vertical skew is left out since it can be constructed using a combination of rotation and scaling

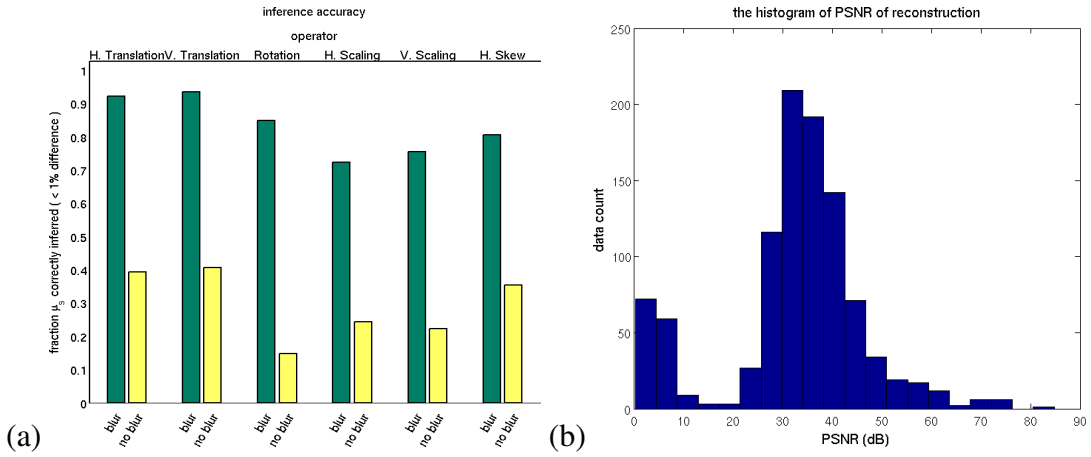

Figure 2: *(a)* The fraction of recovered coefficients which differed by less than 1% from the true coefficient values. Image patches were transformed using a set of hand coded affine transformations (all transformations simultaneously), and recovery was performed via gradient descent of Equation 14. Inference with and without adaptive blurring is compared. *(b)* The distribution of PSNR values for image patches reconstructed using coefficients inferred with adaptive blurring.

The affine transformation operators are derivative operators in the direction of motion. For example, a horizontal translation operator is a derivative operator in the horizontal direction while a rotation operator computes a derivative radially. Our learned operators illustrate this property. Figure 3 shows two of the learned transformation operators, where each 11×11 block corresponds to one column of A and the block's position in the figure corresponds to its pixel location in the original image patch. This can be viewed as an array of basis functions, each one showing how intensity at a given pixel location influences the instantaneous change in pixel intensity at all pixel locations (see Equation 1). In this figure, each basis represents a numerical differentiator. The bottom two rows of Figure 3 show each of the operators being applied to an image patch. An animation of the full set of learned affine operators applied to image patches can be found in the supplementary material.

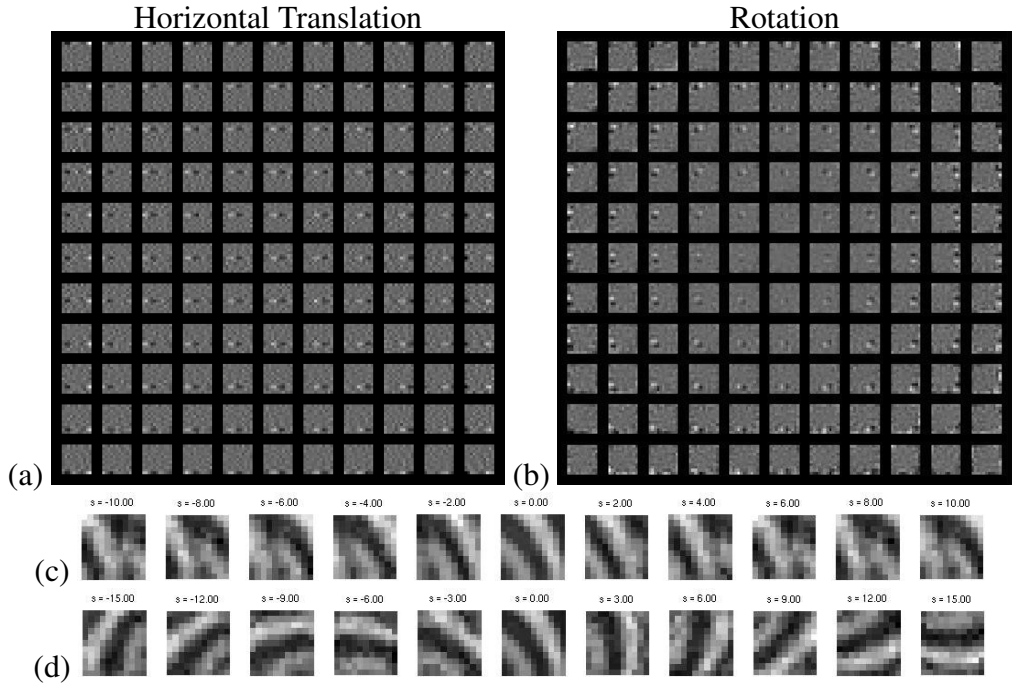

Figure 3: The transformation operators that corresponds to horizontal translation *(a)* and rotation *(b)*. Each 11×11 block corresponds to one column of A and the block's position in the figure corresponds to its pixel location in the original image patch. Each block therefore shows how intensity at one pixel location contributes to the instantaneous change in intensity at other pixel locations. Note that the blocks correspond to spatial derivatives in the direction of motion. Panes *(c)* and *(d)* show the translation and rotation operators, respectively, being applied to an image patch.

7.3 Learning Transformations for Time-Varying Natural Images

To explore the transformation statistics of natural images, we trained the algorithm on pairs of 17×17 image patches cropped from consecutive frames obtained from a corpus of short videos from The BBC's *Animal World Series*. In order to allow the learned transformations to capture image features moving into and out of a patch from the surround, and to allow more direct comparison to motion compensation algorithms, the error function for inference and learning was only applied to the central 9×9 region in each 17×17 patch. Each patch can therefore be viewed as a 9×9 patch equipped with a 4 pixel wide buffer region. In the 15 transformation case, for computational reasons only a 2 pixel wide buffer region was used, so the 15 transformation case acts

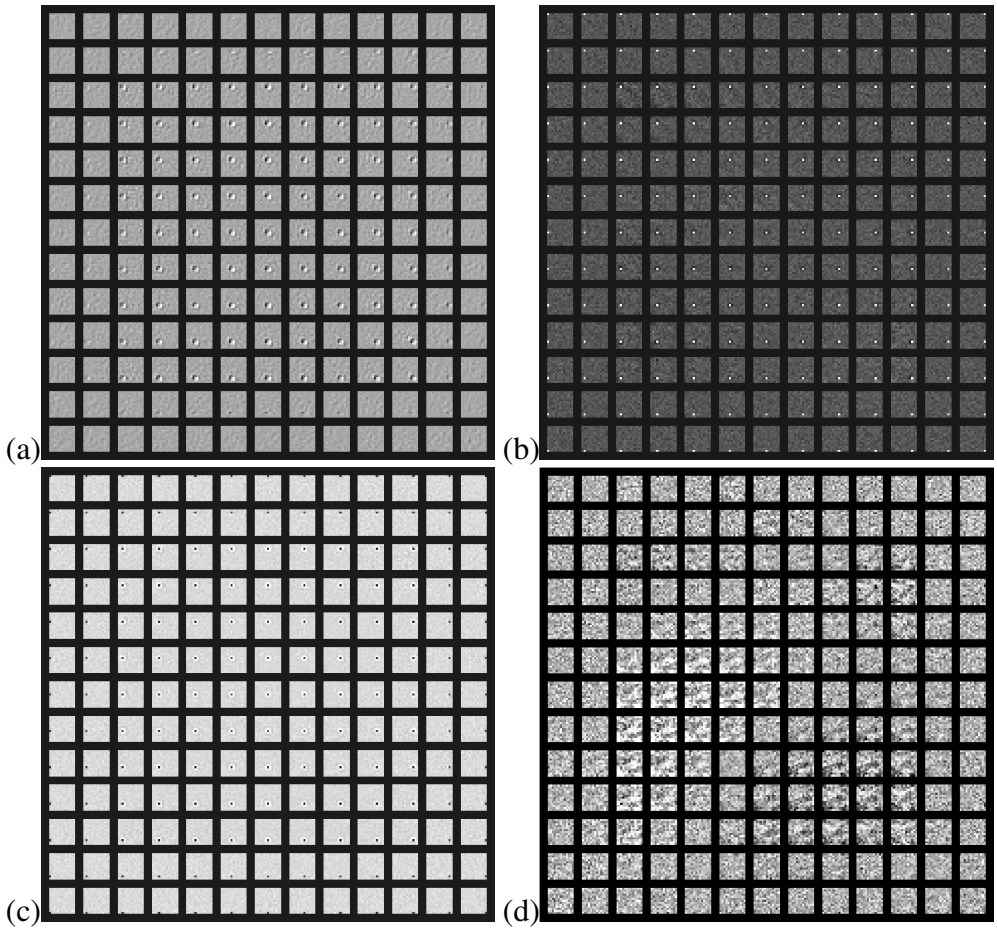

Figure 4: Sample transformation operators from a set of 15 transformations trained in an unsupervised fashion on 13x13 pixel patches (including a 2 pixel buffer region) from natural video. Each 13×13 block corresponds to one column of A and the block's position in the figure corresponds to its pixel location in the original image patch. Each block therefore illustrates the influence a single pixel has on the entire image patch as the transformation is applied. *(a)* is a full field translation operator, *(b)* performs full field intensity scaling, *(c)* performs full field contrast scaling, and *(d)* is difficult to interpret.

on 13x13 pixel patches with the reconstruction penalty on the central 9x9 region.

Training was performed on a variety of models with different numbers of transformations. For several of the models two of the operators were pre-coded to be whole-patch horizontal and vertical translation. This was done since we expect that translation will prove to be the predominant mode of transformation in natural video, and this

allows the algorithm to focus on learning less obvious transformations contained in video with the remaining operators. This belief is supported by the observation that several operators become full field translation when learning is unconstrained, as in the 15 transformation case in Figure 5. Hardcoding translation also provides a useful basis of comparison to existing (whole-patch translation based) motion compensation algorithms used in video compression.

The model case with the greatest variety of transformation operators consisted of 15 unconstrained transformations. A selection of the learned A_k can be found in Figure 4. The learned transformation operators performed full field translations, intensity scaling, contrast scaling, spatially localized mixtures of the preceding 3 transformation types, and a number of transformations with no clear interpretation.

Animations showing a full set of learned operators acting on patches can be found in the supplementary materials.

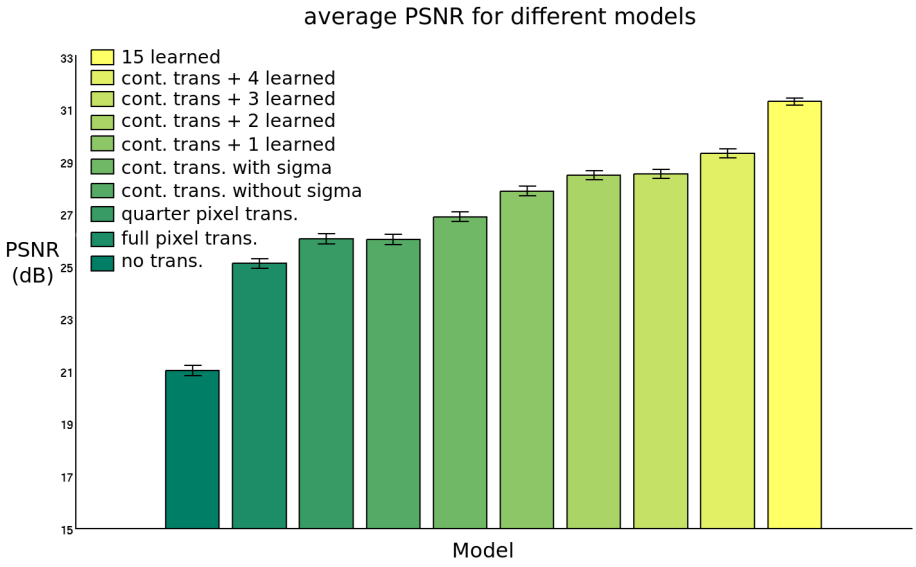

Figure 5: PSNR of the reconstruction of the second frame from 1,000 pairs of frames from natural video using a variety of model configurations.

To demonstrate the effectiveness of the learned transformations at capturing the interframe changes in natural video, the PSNR of the image reconstruction for 1 000 17×17 2 time-step video patches was compared for all of the learned transformation models, as well as to standard motion compensation reconstructions. The models compared were as follows:

1. No transformation. Frame $x^{(t)}$ is compared to frame $x^{(t+1)}$ without any transformation.

2. Full pixel motion compensation. The central 9×9 region of $x^{(t+1)}$ is compared to the best matching 9×9 region in $x^{(t)}$ with full pixel resolution.

3. Quarter pixel motion compensation with bilinear interpolation. The central 9×9 region of $x^{(t+1)}$ is compared to the best matching 9×9 region in $x^{(t)}$ with quarter pixel resolution.

4. Continuous translation without sigma. Only vertical and horizontal translation operators are used in the model, but they are allowed to perform subpixel translations.

5. Continuous translation with sigma. Vertical and horizontal translation operators are used in the model, and in addition adaptive smoothing is used.

6. Continuous translation plus learned operators. Additional transformation operators are randomly initialized and learned in an unsupervised fashion.

7. 15 learned transformation operators. Fifteen operators are randomly initialized and learned in an unsupervised fashion. No operators are hard coded to translation.

7.4 Reconstructing Time-Varying Natural Images

As shown in Figure 5 there is a steady increase in PSNR as the transformation model gets more complex. The use of the learned operators for video compression is explored more fully in Wang et al. [2011].

Conclusions

We have demonstrated and tested a method for learning Lie group operators from image sequences. We made the problem computationally tractable by utilizing an eigendecomposition of the operator matrix, in addition to a transformation specific blurring operator which removed local minima. This model was then applied to recovering

transformation coefficients for affine transformations, learning affine transformations, and to learning the transformations between frames in natural video. These experiments demonstrated that the method was effective and tractable, and that it is able to perform both inference and learning with many transformations on very large inter-frame transformations. We have also demonstrated that the learned models allow better description of the dynamics of natural movies than the standard rigid translation model.

References

D.W. Arathorn. Map-seeking circuits in visual cognition: a computational mechanism for biological and machine vision. 2002.

M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Proc. of the Fourth European Conference on Computer Vision*, pages 320–342, 1996.

C. Cadieu and B.A. Olshausen. Learning transformational invariants from natural movies. *Advances in Neural Information Processing Systems 21*, pages 209–216, 2008.

B.J. Culpepper and B.A. Olshausen. Learning transport operators for image manifolds. *Proceedings of Neural Information Processing Systems 22*, 2010.

W. Einhauser, C. Kayser, P. Konig, and K.P. Kording. Learning the invariance properties of complex cells from their responses to natural stimuli. *European Journal of Neuroscience*, 15(3):475–486, 2002.

E. Kokiopoulou and P. Frossard. Minimum distance between pattern transformation manifolds: Algorithm and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1225–1238, 2009.

Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *Computer Vision and Pattern Recognition*, 2004.

M.S. Lewicki and B.A. Olshausen. Probabilistic framework for the adaptation and

comparison of image codes. *Journal of the Optical Society of America*, 16(7):1587–1601, 1999.

B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging understanding workshop*, pages 121–130, 1981.

R Memisevic and G Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, Jan 2010. URL `http://www.mitpressjournals.org/doi/pdf/10.1162/neco.2010.01-09-953`.

X. Miao and R.P.N. Rao. Learning the lie groups of visual invariance. *Neural Computation*, 19(10):2665–2693, 2007.

B.A. Olshausen, C. Cadieu, B.J. Culpepper, and D. Warland. Bilinear models of natural images. *SPIE Proceedings vol. 6492: Human Vision Electronic Imaging XII (B.E. Rogowitz, T.N. Pappas, S.J. Daly, Eds.)*, 2007.

M. Ortiz, R.A. Radovitzky, and E.A Repetto. The computation of the exponential and logarithmic mappings and their first and second linearizations. *International Journal For Numerical Methods In Engineering*, 52:1431–1441, 2001.

R.P.N Rao and D.L. Ruderman. Learning lie groups for invariant visual perception. *Advances in Neural Information Processing Systems 11*, pages 810–816, 1999.

Mark Schmidt. minfunc. Technical report, http://www.cs.ubc.ca/ schmidtm/Software/minFunc.html, 2009. URL `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`.

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 417–426, 2007.

J. Shen. Resampling volume or image with affine matrix. *http://www.mathworks.com/matlabcentral/fileexchange/21080*, 2008.

N. Vasconcelos and A. Lippman. Multiresolution tangent distance for affine invariant classification. *Proceedings of Neural Information Processing Systems 10*, 1997.

J.D. Victor, Mechler. F., M.A. Repucci, K.P. Purpura, and Sharpee. T. Responses of v1 neurons to two-dimensional hermite functions. *J Neurophysiol*, 95(1):379–400, 2006.

G. Wallis and E.T. Rolls. Invariant face and object recognition in the visual system. *Prog. Neurobiol.*, 51(2):167–194, 1997.

C.M. Wang, J. Sohl-Dickstein, I. Tosic, and B.A. Olshausen. Lie group transformation models for predictive video coding. pages 83–92, 2011.

W. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

A Appendix - Degeneracy in U

We decompose our transformation generator

$$A = V\Lambda V^{-1} \tag{A-1}$$

where Λ is diagonal. We introduce another diagonal matrix R. We can populate the diagonal of R with anything (non-zero) we want, and the following equations will still hold:

$$\begin{aligned} A &= V\Lambda V^{-1} && \text{(A-2)} \\ &= VRR^{-1}\Lambda V^{-1} && \text{(A-3)} \\ &= VR\Lambda R^{-1}V^{-1} && \text{(A-4)} \\ &= (VR)\,\Lambda\,(VR)^{-1} && \text{(A-5)} \end{aligned}$$

If we set

$$U = VR \tag{A-6}$$

then

$$A = U\Lambda U^{-1} \tag{A-7}$$

and R represents a degeneracy in the set of U we are allowed to choose in our decomposition.

We remove this degeneracy in U by choosing R so as to minimize the joint power after every learning step. That is

$$\begin{aligned} R &= \arg\min_R \sum_i \sum_j V_{ij}^2 R_{jj}^2 + \sum_i \sum_j \left(R^{-1}\right)_{jj}^2 \left(V^{-1}\right)_{ji}^2 && \text{(A-8)} \\ &= \arg\min_R \sum_i \sum_j V_{ij}^2 R_{jj}^2 + \sum_i \sum_j \left(V^{-1}\right)_{ji}^2 \frac{1}{R_{jj}^2} && \text{(A-9)} \end{aligned}$$

setting the derivative to 0

$$\begin{aligned} 2\sum_i V_{ij}^2 R_{jj} - 2\sum_i \left(V^{-1}\right)_{ji}^2 \frac{1}{R_{jj}^3} &= 0 && \text{(A-10)} \\ R_{jj} \sum_i V_{ij}^2 &= \frac{1}{R_{jj}^3} \sum_i \left(V^{-1}\right)_{ji}^2 && \text{(A-11)} \\ R_{jj}^4 &= \frac{\sum_i \left(V^{-1}\right)_{ji}^2}{\sum_i V_{ij}^2} && \text{(A-12)} \\ R_{jj} &= \left[\frac{\sum_i \left(V^{-1}\right)_{ji}^2}{\sum_i V_{ij}^2}\right]^{\frac{1}{4}} && \text{(A-13)} \end{aligned}$$

Practically this means that, after every learning step, we set

$$R_{jj} = \left[\frac{\sum_i \left(U^{-1}\right)_{ji}^2}{\sum_i U_{ij}^2}\right]^{\frac{1}{4}} \tag{A-14}$$

and then set

$$U_{new} = UR \tag{A-15}$$

B Appendix - Derivatives

Let

$$\varepsilon = \sum_n (Y_n - Ue^{\mu\Lambda} e^{\frac{1}{2}\Lambda^2\sigma^2} U^{-1} X_n)^2 \tag{B-1}$$

B.1 Derivative for inference with one operator

The learning gradient with respect to μ is

$$\begin{aligned}\frac{\partial \varepsilon}{\partial \mu} &= -\sum_n 2err(n)U\frac{\partial e^{\mu\Lambda}}{\partial \mu}e^{\frac{1}{2}\Lambda^2\sigma^2}U^{-1}X_n \\ &= -\sum_n 2err(n)U\Lambda e^{\mu\Lambda}e^{\frac{1}{2}\Lambda^2\sigma^2}U^{-1}X_n\end{aligned} \tag{B-2}$$

where $err(n)$ is the reconstruction error of the nth sample

$$err(n) = Y_n - Ue^{\mu\Lambda}e^{\frac{1}{2}\Lambda^2\sigma^2}U^{-1}X_n \tag{B-3}$$

Similarly, the learning gradient with respect to σ is

$$\frac{\partial \varepsilon}{\partial \sigma} = -\sum_n 2err(n)Ue^{\mu\Lambda}\sigma\Lambda^2e^{\frac{1}{2}\Lambda^2\sigma^2}U^{-1}X_n \tag{B-4}$$

B.2 Derivative for learning with one operator

The learning gradient with respect to Λ is

$$\frac{\partial \varepsilon}{\partial \Lambda} = -\sum_n 2err(n)U\frac{\partial e^{\mu\Lambda}e^{\frac{1}{2}\Lambda^2\sigma^2}}{\partial \Lambda}U^{-1}X_n \tag{B-5}$$

It is easy to see then this derivative with respect to each element in Λ is

$$\frac{\partial \varepsilon}{\partial \lambda_k} = -\sum_n 2err(n)U(\mu e^{\mu\lambda_k} + \sigma^2\lambda_k e^{\frac{1}{2}\Lambda^2\sigma^2})\,U^{-1}X_n \tag{B-6}$$

Therefore, in matrix form, the derivative is

$$\frac{\partial \varepsilon}{\partial \Lambda} = -\sum_n 2err(n)U(\mu e^{\mu\Lambda} + \sigma^2\Lambda e^{\frac{1}{2}\Lambda^2\sigma^2})\,U^{-1}X_n \tag{B-7}$$

The learning gradient with respect to U is

$$\begin{aligned}\frac{\partial \varepsilon}{\partial U} &= -\sum_n 2err(n)\frac{\partial U}{\partial U}e^{\mu\Lambda}e^{\frac{1}{2}\Lambda^2\sigma^2}U^{-1}X_n \\ &\quad -\sum_n 2err(n)Ue^{\mu\Lambda}e^{\frac{1}{2}\Lambda^2\sigma^2}\frac{\partial U^{-1}}{\partial U}X_n\end{aligned} \tag{B-8}$$

Recall that

$$\frac{dU^{-1}}{U} = -U^{-1}\frac{dU}{dU}U^{-1} \tag{B-9}$$

The learning gradient is hence

$$\begin{aligned}\frac{\partial \varepsilon}{\partial U} = &-\sum_n 2err(n)e^{\mu\Lambda}e^{\frac{1}{2}\Lambda^2\sigma^2}U^{-1}X_n \\ &+\sum_n 2err(n)Ue^{\mu\Lambda}e^{\frac{1}{2}\Lambda^2\sigma^2}U^{-1}U^{-1}X_n\end{aligned} \tag{B-10}$$

B.3 Derivative for Complex Variables

To accommodate for the complex variables U and Λ, we rewrite our objective function as

$$\varepsilon = \sum_n err(n)^T\overline{err(n)} \tag{B-11}$$

where $\overline{err(n)}$ denotes the complex conjugate. The derivative of this error function with respect to any complex variable can be then broken into the real and and imaginary part

$$\begin{aligned}\frac{\partial \varepsilon}{\partial \Lambda} &= \sum_n err(n)^T\overline{\frac{\partial err(n)}{\partial \Lambda}} + \frac{\partial err(n)^T}{\partial \Lambda}\overline{err(n)} \\ &= \sum_n err(n)^T\overline{\left(\frac{\partial err(n)}{\partial \Lambda}\right)} + \left(\frac{\partial err(n)}{\partial \Lambda}\right)^T\overline{err(n)} \\ \Re\left\{\frac{\partial \varepsilon}{\partial \Lambda}\right\} &= 2\Re\left\{\sum_n err(n)^T\overline{\left(\frac{\partial err(n)}{\partial \Lambda}\right)}\right\} \\ \Im\left\{\frac{\partial \varepsilon}{\partial \Lambda}\right\} &= -2\Im\left\{\sum_n err(n)^T\overline{\left(\frac{\partial err(n)}{\partial \Lambda}\right)}\right\}\end{aligned} \tag{B-12}$$

B.4 Derivatives in Matrix Notation

For completeness, we can write the derivatives in matrix notation as follows.

$$\begin{aligned}
\frac{\partial \varepsilon}{\partial \mu} &= -2E^T U \Lambda K U^{-1} X \\
\frac{\partial \varepsilon}{\partial \sigma} &= -2\sigma E^T U \Lambda^2 K U^{-1} X \\
\frac{\partial \varepsilon}{\partial \Lambda} &= -2E^T (\mu e^{\mu\Lambda} + \sigma^2 \Lambda e^{\frac{1}{2}\Lambda^2\sigma^2})\, U^{-1} X \\
\frac{\partial \varepsilon}{\partial U} &= -2E^T K U^{-1} X + 2E^T U K U^{-1} U^{-1} X
\end{aligned} \tag{B-13}$$

where

$$K = e^{\mu\Lambda} e^{\frac{1}{2}\Lambda^2\sigma^2} \tag{B-14}$$

E and X are matrices with columns of err(n) and X_n respectively.

B.5 Derivatives for Manifold Penalty

We have a model

$$\dot{I} = AI \tag{B-15}$$

with solution

$$I_1 = e^{As} I_0. \tag{B-16}$$

We want to find and minimize the distance traveled by the image patch I_0 to I_1 under the transformation operator A. The total distance is

$$d = \int_{t=0}^{s} ||\dot{I}||_2^1 dt. \tag{B-17}$$

This then gives the following,

$$\begin{aligned}
d &= \int_{t=0}^{s} ||Ax^{(t)}||_2^1 dt \\
&= \int_{t=0}^{s} ||Ae^{At} I_0||_2^1 dt \\
&= \int_{t=0}^{s} \sqrt{(Ae^{At} I_0)^T (Ae^{At} I_0)} dt \\
&= \int_{t=0}^{s} \sqrt{I_0^T (e^{At})^T A^T A e^{At} I_0} dt
\end{aligned} . \tag{B-18}$$

We don't know how to solve this analytically. Instead we make the approximation

$$\int_{t=0}^{s} ||\dot{I}||_2 dt \simeq s||AI_0||_2, \tag{B-19}$$

with derivatives

$$\begin{aligned} d &= s(I_0^T A^T A I_0)^{\frac{1}{2}} \\ \frac{\partial d}{\partial s} &= (I_0^T A^T A I_0)^{\frac{1}{2}} \\ \frac{\partial d}{\partial A} &= s\frac{1}{2}(I_0^T A^T A I_0)^{-\frac{1}{2}} 2 A I_0 I_0^T \end{aligned} \quad . \tag{B-20}$$

Second approximation

$$\begin{aligned} \int_{t=0}^{s} ||\dot{I}||_2 dt &\simeq s||AI_{\frac{s}{2}}||_2 \\ &= s||Ae^{A\frac{s}{2}} I_0||_2 \end{aligned} \tag{B-21}$$

The derivative is therefore

$$\begin{aligned} d &= s\left(I_0^T (e^{A\frac{s}{2}})^T A^T A e^{A\frac{s}{2}} I_0\right) \\ &= \frac{\partial s}{\partial s}\left(I_0^T (e^{A\frac{s}{2}})^T A^T A e^{A\frac{s}{2}} I_0\right) + I_0^T \frac{\partial (e^{A\frac{s}{2}})^T}{\partial s} A^T A e^{A\frac{s}{2}} I_0 + I_0^T (e^{A\frac{s}{2}})^T A^T A \frac{\partial e^{A\frac{s}{2}}}{\partial s} I_0 \\ &= B + 2sB \end{aligned} \tag{B-22}$$

where $B = I_0^T (e^{A\frac{s}{2}})^T A^T A e^{A\frac{s}{2}} I_0$

We can check this approximation against numerical integrals